Context Interchange:
A Lattice Based Approach

M.P. Reddy
Amar Gupta

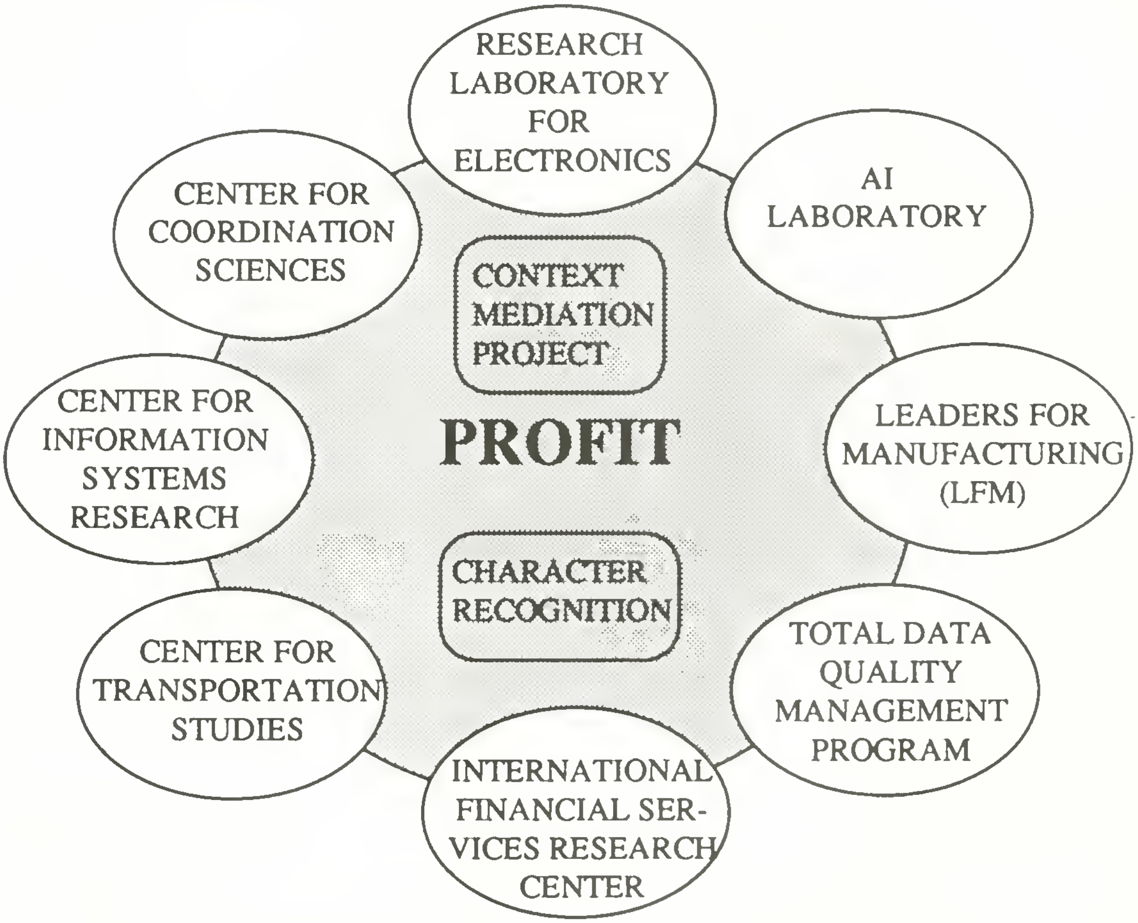
WP #3780 August 1994
PROFIT #94-19

Productivity From Information Technology
"PROFIT" Research Initiative
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
(617)253-8584
Fax: (617)258-7579

The research described herein has been supported (in whole or in part) by the Productivity From Information Technology (PROFIT) Research Initiative at MIT.

Productivity From Information Technology (PROFIT)

The Productivity From Information Technology (PROFIT) Initiative was established on October 23, 1992 by MIT President Charles Vest and Provost Mark Wrighton "to study the use of information technology in both the private and public sectors and to enhance productivity in areas ranging from finance to transportation, and from manufacturing to telecommunications." At the time of its inception, PROFIT took over the Composite Information Systems Laboratory and Handwritten Character Recognition Laboratory. These two laboratories are now involved in research related to context mediation and imaging respectively.


RESEARCH LABORATORY FOR ELECTRONICS
AI LABORATORY
LEADERS FOR MANUFACTURING (LFM)
TOTAL DATA QUALITY MANAGEMENT PROGRAM
INTERNATIONAL FINANCIAL SERVICES RESEARCH CENTER
CENTER FOR TRANSPORTATION STUDIES
CENTER FOR INFORMATION SYSTEMS RESEARCH
CENTER FOR COORDINATION SCIENCES
CONTEXT MEDIATION PROJECT
PROFIT
CHARACTER RECOGNITION

In addition, PROFIT has undertaken joint efforts with a number of research centers, laboratories, and programs at MIT, and the results of these efforts are documented in Discussion Papers published by PROFIT and/or the collaborating MIT entity.

Correspondence can be addressed to:

The "PROFIT" Initiative
Room E53-310, MIT
50 Memorial Drive
Cambridge, MA 02142-1247
Tel: (617) 253-8584
Fax: (617) 258-7579
E-Mail: profit@mit.edu

Context Interchange: A Lattice Based Approach

M. P. Reddy* and A. Gupta
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

The level of semantic data interoperability between a source and a receiver is a function of the context interchange mechanism that operates between the source and the receiver. The semantic interoperability mechanisms in existing systems are usually static in nature and cannot cope up with changes in the semantics of data either at the source or at the receiver. In this paper, we propose a context interchange mechanism, based on lattice theory which can handle changes in the semantics of data at both the source and the receiver. A site-copy selection algorithm is also presented in this paper which selects the set of sources that can supply semantically meaningful data to the query of a particular source.

Keywords: semantic-interoperability, context-interchange, lattice theory, and query-processing.

1 Introduction

It would not be an exaggeration to claim that data required for any application are available at some data-source in the world. Data in such data-sources might have different context than the application context. If it is possible to convert the context of data in the data-source to the context of an application, then it is less expensive to reuse these data than collecting the required data for an application from the scratch. Research on data communications has

* Dr. Reddy is now with Kenan Systems Corporation, One Main Street, Cambridge, MA 02142-1517; e-mail:preddy@kenan.com

concentrated on the physical obstacles to the reliable transfer of data between a source and a receiver, but rarely on issues such as the mismatch between the context of data-source and the context of the data-receiver.

What exactly constitutes the context is difficult to answer [Lyo81]. The concept of context has been addressed in many areas such as sensory process, perception, language, concept learning, recall and recognition [Bur52, Coe77, Tho88]. The main reason for the context assuming a central role in these areas is that objects and their associated events constitute an integral part of their environment and cannot be understood in isolation of that environment. In this paper we do not attempt to give precise definition for this term, even though this is part of our long term research objective. We assume that context knowledge of a data item is a triple given by the semantic knowledge of the data, the organization of the data, and the quality parameters of the data. In this paper, we concentrate only on the semantic component of the context, which is formally defined in Section 3.

Consider the process by which a financial analyst accesses the prices for shares of a particular company. He or she needs to gather information from several stock exchanges located in different nations and must overcome semantic discrepencies at multiple levels: the stock prices are stated in different currencies, the currencies are floating with respect to each other; the stock price may be the latest-price or the closing-price; etc. Such semantics are implicit in many existing databases. Unless these semantics are made explicit, it is difficult to identify and resolve underlying semantic incompatibilities. The fundamental question is how to make such semantics explicit and how to quickly identify the incompatibilities and resolve them if possible.

A number of researchers [LR82, Tem89, DAT87, Ke88, RJPS89, BT84] have proposed solutions that aim to identify semantic incompatibilities during the process of schema integration. In this scenario, every application defines its own views on the integrated schema and these views are used to translate the semantics of the integrated schema into application semantics. In this approach, the semantics of data in a database are first translated into the semantics of the integrated schema and then translated into the semantics of the application. This strategy is expensive if any changes occurs in the semantics of the data that is being integrated. In [SM91], a rule-based approach was proposed in which these semantic differences are dynamically identified and resolved using the context information associated with the data items required by an application. In this approach, the semantics of data in a database can be directly translated into the semantics of the application. In this paper, we extend this model with a lattice-based representation for the context knowledge. We believe that the lattice representation is more natural for representing the context knowledge and for cross comparison.

Our context interchange scenario is shown in Figure 1. This scenario consists of a set of applications, a set of data sources, and a context mediator. Each application or data-source is associated with a Context Data Repository (CDR), which explicitly specifies the context of each concept relevant to it from its point of view. Whenever an application requests for data, the context mediator ensures that the application receives semantically meaningful data from various data sources (if it is possible). As such, the context mediator must possess the capability to reason with different contexts (i.e., with different CDRs) simultaneously. The representation scheme of context in each CDR can enable the context mediator to simultaneously reason with different CDRs. This imposes a need for systematic description of context knowledge of application requirements and the context knowledge of data-sources; we have adopted the lattice model for such representation based on the following considerations:

- A database can supply meaningful data to an application provided the database context is *more general than* the application context. The *more general than* relation can be elegantly modeled using the Lattice model. In a lattice, the context becomes more specific as one moves upward and more generalized as one moves downward.

- Context knowledge can be economically represented in the form of a lattice as the knowledge present at a particular context, which can be shared by all contexts which are more specific than that context.

- Reasoning with context knowledge in the form of a lattice is economical because an application receives meaningful solution from all data sources which are in a more generalized context than the application, and the more general than relation is easy to trace in a lattice.

Within the lattice based context interchange framework, conditions under which a data source can supply semantically meaningful data to address an application's data requirements are derived in this paper. We have proposed a context driven site-copy selection algorithm, which selects various candidate data-sources that can supply meaningful data to queries initiated by different applications.

This paper is organized as follows: Section 2 presents a brief overview of representation schemes for the context knowledge. The approach selected in this paper for representing context knowledge is presented in Section 3. In Section 4, a mechanism to describe context data repository for databases and their applications is discussed. A procedure for context mediation is presented in Section 5. Query processing and associated issues are discussed in Section 6. The last section contains our conclusions.

2 Lattice-based Representation of Context

The concept of context has been treated from different perspective by various researchers. For example, the Truth Maintenance System (TMS) [Doy89] and the Assumption-based Truth Maintenance System (ATMS)[dK86a] have adopted different representation schemes for representing the notion of context. In TMS, each datum is labeled either IN or OUT, as determined by the given boundary condition. The notion of context is implicit in the boundary conditions and all the data items which are believed to be true in that context are labelled IN and all the data items which are not believed to be true in that context are labelled OUT. The context changes only when one of the boundary conditions changes. As such, TMS maintains only one context (global context) at any given point of time. In contrast, ATMS provides all the contexts in which the data item are believed and can represent multiple contexts simultaneously. The computational advantages of ATMS are discussed in [McD83, dK86a, dK86b, dK86c]. The context interchange problem is somewhat closer to that of ATMS as the context-mediator needs to interact and reason with multiple

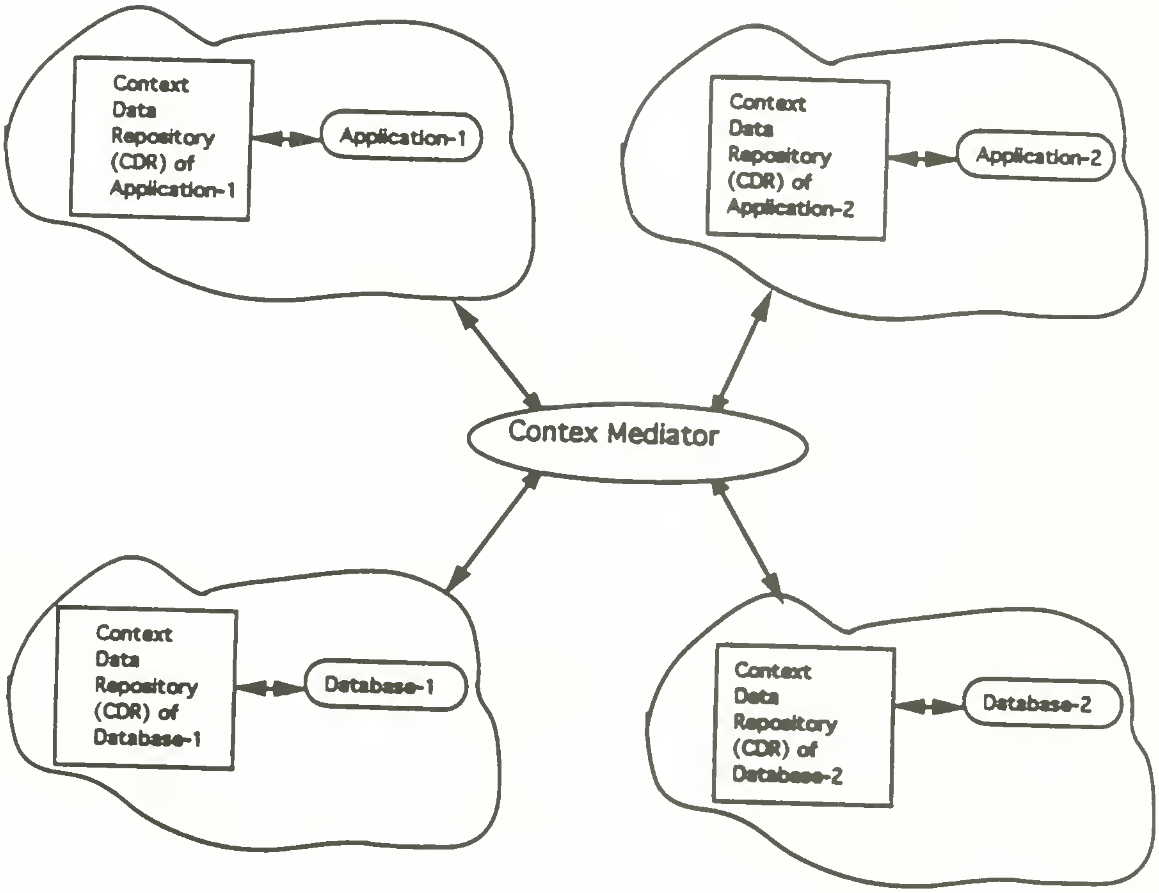

Context Data Repository (CDR) of Application-1
Application-1
Context Data Repository (CDR) of Application-2
Application-2
Contex Mediator
Context Data Repository (CDR) of Database-1
Database-1
Context Data Repository (CDR) of Database-2
Database-2

Figure 1: Context Interchange Scenario

contexts simultaneously.

2.1 Notation and Definitions

Let X and Y represent the context knowledge about a concept from two different viewpoints. Five possible relations can exist among X and Y, as follows:

(i) X = Y ; X and Y are in the same context.

(ii) $X \subset Y$; X is a more generalized context than Y.

(iii) $X \supset Y$; X is a more specific context than Y.

(iv) $X \cap Y \neq \phi$ and (i), (ii), or (iii) are not satisfied; X and Y possess some context in common.

(v) $X \cap Y = \phi$; X and Y are totally disjoint contexts.

Two contexts are comparable if either (i), (ii) or (iii) is satisfied. Therefore one can define only partial order among the set of contexts using the relation $\subseteq$ and form a context-lattice. A lattice is a partially ordered set, with $X \subseteq Y$ meaning $X \cap Y = X$ and $X \cup Y = Y$, in which each pair of elements possesses a greatest lower bound and a least upper bound within the set. If $X \subseteq Y$ then one says that X is as general context as context Y [Sho91].

A context alone is not interesting; it is interesting only when it is linked with all assertions which are true in that context. Contexts can be viewed as envelopes enclosing some assertions. If an assertion A is true in the context X then this information is represented as A^X. This assertion may be true in one context and untrue in another context. If A^X is valid and if $Y \subseteq X$ then A^Y is also valid. In other words, if an assertion A is true in a particular context, say X, then it is true in all contexts which are more general than X. This is the basis for the context interchange mechanism presented in this paper.

3 Semantic Assertions and Context-Knowledge

Each data-source is visualized as a set of distinct object types and each object is an aggregation of a set of properties. We categorize properties into two classes: primitive properties and non-primitive properties. Semantics of primitive properties are the same across all applications and data sources, whereas the semantics of a non-primitive property may be different for different applications and at different data sources. The semantics of a non-primitive property are captured through a context-assertion lattice.

The context-assertion domain of a property is the set of contexts relevant to the property and a set of assertions which may be true in each context. The context-assertion domain of a property is given by the context-assertion lattice of the property. This lattice is constructed from the context lattice and the semantic-assertions domain. The context lattice, the semantic assertion domain, and the context assertion lattice are described in the following subsections.

3.1 Context-Lattice

As mentioned earlier, the semantics of any property cannot be understood in isolation of the environment/context in which it is intended to be used. In other words, the environment/context associated with a property functionally determines the semantics of the property. As observed in [SSR92], the environment of a property can be occasionally determined by other properties of the object. These properties are collectively referred to as the environment schema of the non-primitive property. The environment schema needs to be tagged whenever a non-primitive property is moved from one environment to another. The set of all possible environments/contexts in which the property is defined is called the context domain of the property. The context domain of a property is represented by the context lattice.

Let Environment(P) denote the environment scheme of a non-primitive property P. Let E_j be a property in Environment(P). Let Dom(E_j) be the domain or the set of legal values associated with E_j. The assignment of each $e_i \in$ Dom(E_j) to the property E_j sets a different context for the non-primitive property P. For example, let Dom(Instrument-type) be { Equity, Future } and Dom(Exchange) be {Nyse, Tokyo }. Assignment of Equity to Instrument-type sets a particular context to property Trade-price, whereas the assignment of Future to Instrument-type sets another context to the property Trade-price. Different contexts associated with the property Trade-price in the example are shown in Table-1.

Table-1	
Notation	Context
A	Instrument-type='Equity'
B	Instrument-type='Future'
C	Exchange='Nyse'
D	Exchange='Tokyo'

We define the base context set, denoted by R, for a non-primitive property P, as follows:

$$R = \bigcup_{E_j \in \text{Environment}(P)} Dom(E_j)$$

The powerset of R generates a context-lattice for the property P.

In the example, R is given by the set { A, B, C, D }. The power set of R generates the context-lattice for Trade-price. The context-lattice generated for the Trade-price is shown in Figure 2. Crossed out nodes in the context-lattice are called no-good nodes. In other words such contexts are not applicable to the particular property.

As mentioned earlier, contexts are interesting only when they are linked with all assertions which are true in that context. The issue of semantic assertions of a non-primitive property is discussed in the next subsection.

3.2 Semantic Assertions

The semantic-assertions of each property can be defined using a finite set of parameters called meta-properties. Consider a non-primitive property P. Let Meta-Properties(P) be the set of meta-properties associated with the property P. For example, the property Trade-price may have two meta-properties 'Currency' and 'Status'.

Meta-Properties(Trade-price) = { Currency, Status }

Each meta-property is associated with a finite set of meta-values. Let MP_i be a meta property of property P. Let Meta-Values(MP_i) be the set of meta values associated with the meta-property MP_i. For example, the meta-property 'Currency' may have two meta-values 'Dollars' and 'Yens'.

Meta-Values (Currency) = { Dollars, Yens }

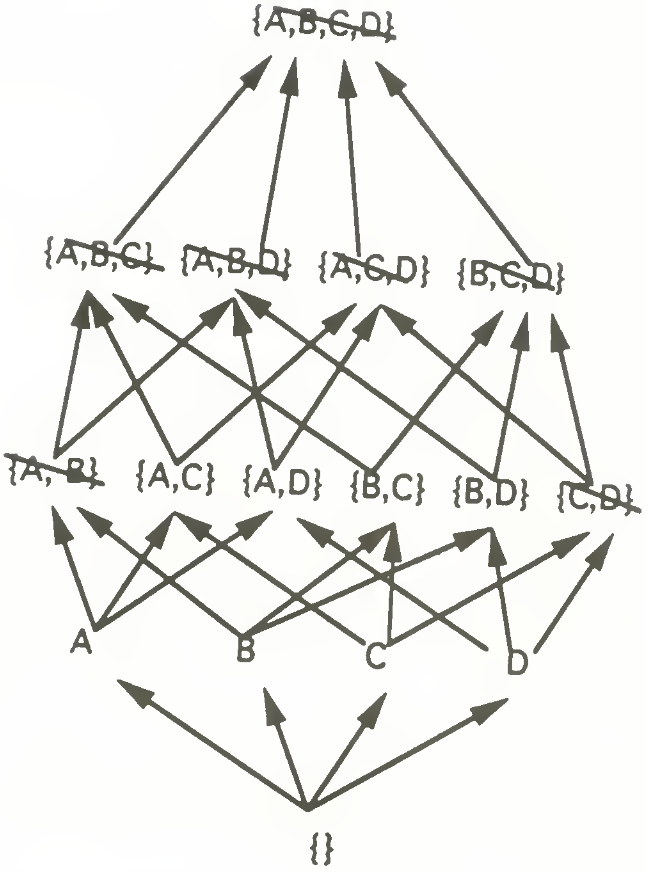


Figure 2: Context-lattice of Trade-price (Refer to Table-1 for definitions of A through D)

In some context, say 'X', the meta-property MP_i may have a meta-value MV_1; in another context, it may have a meta-value MV_2 and so on. For example, the meta-value of currency may be Dollars in one context and Yens in another context. This implies that two semantic assertions are possible for the meta-property Currency. Similarly, if one assumes that Meta-Values(Status) is given by { 'Latest-price', 'Closing-price' }, then semantic assertions for 'Trade-price' will be as shown in Table-2.

Table-2

Notation	Semantic Assertions
E	Currency='Dollars'
F	Currency='Yens'
G	Status='Latest-price'
H	Status='Closing-price'

Let the set of semantic-assertions associated with the meta property MP_i be denoted by Assertions(MP_i). The semantic assertion domain of property P is defined by the cross product of semantic assertions associated with each meta-property of P. Let N be the number of meta-properties associated with the property P. The semantic assertion domain of P can be formally written as:

Semantic-Assertion- Domain(P) =

$$\text{Assertions}(MP_1) \ \text{x} \ \text{Assertions}(MP_2) \ \text{x} \ldots \text{x} \ \text{Assertions}(MP_N)$$

In the example, the semantic-assertion domain of the property 'Trade-price' is given by:

Semantic-Assertion-Domain(Trade-price)

= Assertions(Instrument-type) x Assertions(Exchange)

= { E, F } X { G, H }

= {EG, EH, FG, FH }

= { EG, EH, FG, FH }

In the above expression, EG means E $\wedge$ G, where $\wedge$ is the boolean 'and' operator; similar notation holds for other terms. If two semantic assertions, say E and F are convertible, then EH and FH are also convertible. For example, if Trade-price in Dollars is convertible to a Trade-price in Yens then a Trade-price which is Latest-price in Dollars is also convertible to a Trade-price which is Latest-price in Yens.

3.3 Context-Assertion Lattice of a Non-primitive Property

The context-assertion lattice of a property is generated by the cross-product of the context-lattice and the semantic assertion domain. Therefore, the context-assertion lattice couples the contexts of a property with its associated assertions. As an example, the context-assertion lattice of Trade-price is as shown in Figure 3.

A node n_i in the context-assertion lattice is a pair denoted by (x_i, y_i). The first co-ordinate projection (denoted by PJ_1) is called the context co-ordinate and the second co-ordinate projection (denoted by PJ_2) is called the semantic co-ordinate. These two co-ordinates are defined as follows:

$PJ_1(n_i) \rightarrow x_i$

$PJ_2(n_i) \rightarrow y_i$

The relations $\subseteq$ and $\supset$ among two nodes n_a and n_d in the context-lattice are defined as follows:

$n_a \subseteq n_d$ provided $PJ_1(n_a) \subseteq PJ_1(n_d)$ and $PJ_2(n_a) = PJ_2(n_d)$.

$n_a \supset n_d$ provided $PJ_1(n_a) \supset PJ_1(n_d)$ and $PJ_2(n_a) = PJ_2(n_d)$.

Given any two nodes, n_a and n_d in the context-assertion lattice, one can define three relations as follows:

Total context-lift: The context of node n_d can be totally lifted to the context of node n_a, provided $n_d \subseteq n_a$.

Partial context-lift: The context of node n_d can be partially lifted to the context of a node n_a, whenever $n_d \supset n_a$.

No context-lift: The context of node n_d cannot be lifted to another node n_a, if the context of n_d cannot be either totally or partially lifted to the context of n_a.

Node Equivalence: Given a node n_d in the context-assertion lattice, we define a node-equivalence class denoted by $[n_d]$ as follows:

$[n_d] = \{ n_a \mid n_a$ is in the context-assertion lattice and $PJ_1(n_d) = PJ_1(n_a)$ and $PJ_2(n_d)$ is convertible to $PJ_1(n_a) \}$

If the context of any node $n_k \in [\, n_d]$ can be lifted to the context of n_i then every node $n_l \in [\, n_d \,]$ can be lifted to the context of n_i.

The context and its associated semantic assertions of any instance of P can be given by one of the nodes of the context-assertion lattice. Our model requires a context-assertion lattice for each non-primitive property that is being shared (interoperable) among applications and data-sources. The use of context-assertion lattice in describing the context knowledge of a data-source or an application is discussed in the next section.

4 Context Definition Repository

In the proposed context interchange architecture, each data-source or application needs to have a Context Definition Repository(CDR). Semantic assertions and their associated contexts for each and every non-primitive property of a data source collectively constitute the CDR for the data-source. Similar definition holds for an application's CDR.

Consider a database, db1, which supplies the instance values of the property Trade-price. Assume that the instances of Trade-price are defined in three different contexts, namely C1, C2, and C3. The context C1 is defined by Instrument-type = 'Equity' and Exchange = 'Nyse', the context C2 is defined by Exchange = 'Tokyo', and the context C3 is defined by Instrument-type = 'Future'. Let the semantic assertions Status= 'latest-price' and Currency = 'Dollars' are true in context C1, and Status = 'Closing-price' and Currency = 'Yens' are true in C2, and finally Status = 'Closing-price' and Currency= 'Dollars' are true in context C3. These contexts C1, C2, C3 and their respective semantic assertions can be mapped to nodes n21, n20, and n9 respectively in the context assertion lattice of property Trade-price. The collection of all contexts and semantic assertions which are true in these contexts for a non-primitive property in a particular data-source forms the characterizing environment for the non-primitive property at that particular data-source. Now one can define the characterizing environment of the Trade-price in the data-source db1 as:

$\gamma^{db1}_{Trade-price} = \{n21,\ n20,\ n9\}$

As such, the context data repository of a data-source can be described by defining the

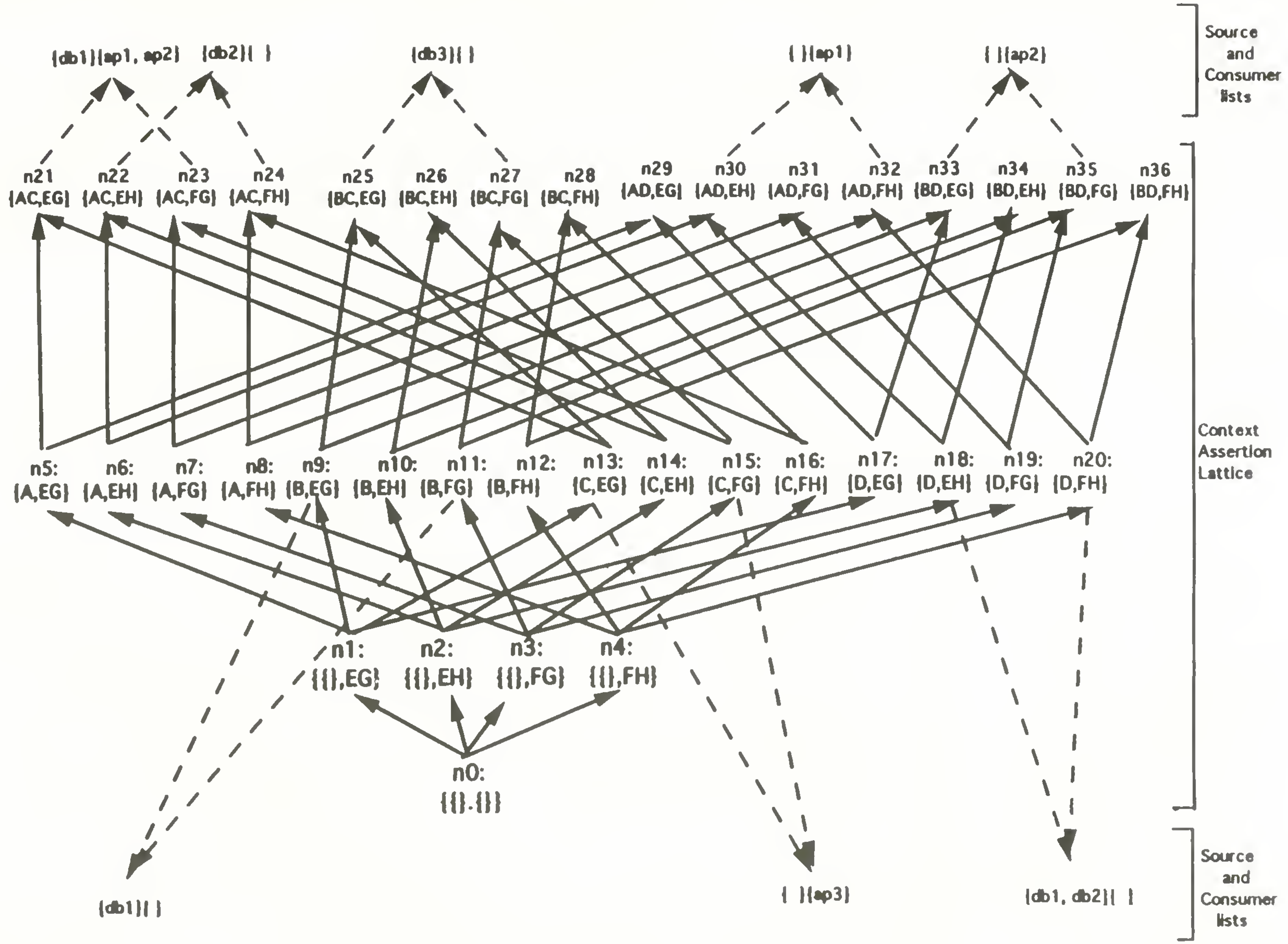


Figure 3: Context-assertion Lattice of Trade-price

characterizing environments for each and every non-primitive property that can be supplied. The collection of such characterizing environments, one for each non-primitive property relevant to a data-source forms the Context Definition Repository (CDR) for that data-source. Similarly, the characterizing environments for each and every non-primitive property required for the application constitute the Context Definition Repository (CDR) for that application.

5 Context Comparison

One aim of context comparison is to ensure that applications receive only meaningful data from data-sources. To accomplish this, the query processor must analyze the query, identify all the relevant nodes in the CDR of the application, compare these nodes with the nodes in the CDRs' of data sources, and identify potential sources which can supply meaningful data to the application. Checking for these relations at run-time, that is during the query evaluation time implies more burden on the query processor. Therefore, the context comparison task is ideally delegated to context-mediator. The context-mediator can make certain comparisons in advance and the query-processor can utilize the results of these comparisons at query evaluation time.

In this scenario, the goal of context-mediator can be conceptually outlined. The context-mediator is provided with a set of CDRs of databases and applications. The task of the context-mediator is to efficiently determine the subset of databases whose context can be lifted to an application's context. Efficiency can be achieved by taking advantage of two important factors. First, the context-mediator can be supplied with changes in the CDR of the relevant application or database, allowing the context-mediation process to be incremental, with updates for the changed contexts only. Second, the query-processor usually needs to know only which data-source can supply meaningful data to an application's query, and only rarely the contents of CDR of a data-source. The context mediator can therefore be constructed as an intermediate data structure which can make very rapid comparisons of contexts. This data structure must also help the context-mediator in identifying and correcting the pre-compiled comparisons, whenever changes are made to the existing CDRs.

The context-mediator associates every node in the context-assertion lattice with two lists: a **Source-list** and a **Consumer-list**. A Source-list states all the databases which contain the same node in their respective CDRs. An consumer-list states all the applications which are having the same node in their respective CDRs. These lists are the same for all equivalent nodes in the context-assertion lattice.

To demonstrate context comparison, consider three applications ap1, ap2, and ap3, whose data requirements span over data-sources db1, db2, and db2. Assume that applications ap1, ap2, and ap3 require the instances of the non-primitive property 'Trade-price' in different contexts and further that 'Trade-price' is available in db1, db2, and db3 in different contexts. The semantics of 'Trade-price' in different applications and in different databases are given by characterizing environments in their respective CDRs.

$\gamma^{db1}_{Trade-price} = \{n21,\ n20,\ n9\}$

$\gamma^{db2}_{Trade-price} = \{n20,\ n22\}$

$\gamma^{db3}_{Trade-price} = \{n25\}$

$\gamma^{ap1}_{Trade-price} = \{n21,\ n32\}$

$\gamma^{ap2}_{Trade-price} = \{n21,\ n33\}$

$\gamma^{ap3}_{Trade-price} = \{n13\}$

Consider the context-assertion lattice for 'Trade-price' shown in Figure 3. Using the above given characterizing environments of 'Trade-price', source-list and consumer-list for each node in the context-assertion lattice can be constructed. As an example, consider node n21 in the context-assertion lattice of 'Trade-price'. The element db1 is entered into the Source-list and elements ap1 and ap2 are entered into the Consumer-list of the node n21. The same source-list and consumer-list will be attached to all nodes which are equivalent to n21, i.e., nodes like n23. In Figure 3, for each node, an arrow that is leaving the node with broken line shows the source-list and the consumer-list associated with the node. For simplicity, only non-empty source and consumer lists are shown in this figure.

With the help of Source-list and Consumer-list, the context mediator can identify potential sources which can provide semantically meaningful solution to an application's query.

The context mediator will become a performance bottleneck if every application needs to consult the context mediator for each of its queries. To avoid this performance bottleneck, the context-mediator is envisaged to distribute the required knowledge of context mediation among CDRs' of applications. Every node in the CDR of an application is augmented with two disjoint lists, namely Total-Suppliers-List and Partial-Suppliers-List. The derivation of Total-Suppliers-List and Partial-Suppliers-List is discussed in the following paragraphs.

Total-Suppliers-lists: The total-supplier-list contains all the data-source nodes whose context can be totally lifted to the context of an application node. The union of Source-lists associated with each node present in each chain and reaching a particular application node in the context assertion lattice forms the Total-suppliers-list for the application node. These chains also include the application node

Partial-Suppliers-list: The Partial-Supplier-List gives all the data-source nodes which can only be partially lifted to the application node. The union of Source-lists of each node present in each chain leaving the application node in the context assertion lattice forms the Partial-Support-List for the application node. These chains exclude the application node.

Consider node n21 in the CDR of application ap1. There are two chains reaching this node in the context assertion lattice, namely {n21, n13, n1, n0} and {n21, n5, n1, n0}. The union of source-lists of all nodes in these chains is { db1 }. Therefore the Total-Suppliers-List of n21 of application ap1 will be { db1 }. Since there are no chains leaving n21, the Partial-Support-List of n21 in the CDR of ap1 will be empty. Similarly, for node n32, there are two chains reaching this node, namely { n32, n20, n4, n0 }. Therefore the Total-Support-List of n32 will be { db1, db2 }, and since there are no chains leaving n21, the Partial-Suppliers-List will be empty.

In this scenario, an application can determine the list of potential sources which can supply semantically meaningful solution to its query.

$\gamma^{ap1}_{Trade-price} = \{(n21\{db1\}\{\}),\ (n32\{db1, db2\}\{\})\}$

If the application needs data related to 'Trade-price' in the context n21 then it will select db1, however, if the 'Trade-price' data needed are in the context n32 then either db1 or db2 is selected.

Similarly, one can write the characterizing environments of Trade-price in ap1 and ap2 with Total-Suppliers-lists and Partial-Suppliers-lists.

$\gamma^{ap2}_{Trade-price} = \{(n22\{db1\}\{\}),\ (n34\{\ db1\}\{\ \})\}\{db1\}$

$\gamma^{ap3}_{Trade-price} = \{(n14\{\}\{db1, db3\})\}\{\}$

From the above characterizing environments, applications ap2 and ap3 can select data-sources which are compatible to the required context of the property 'Trade-price'.

The issue that still needs to be addressed is the process of ensuring the correctness of Total-Suppliers-list and Partial-Suppliers-lists that are distributed among CDRs of several applications, in the event of changes in semantics either at data source level or at application level. This issue is addressed in the next subsection.

5.1 Integrity of Semantic Mappings

Semantics of a non-primitive property can change either at an application or at a data source. A mechanism to cope with changes in the semantics of a non-primitive property in the proposed context interchange architecture is discussed in the following paragraphs.

Changes in Application Data Semantic: Whenever the semantics of a non primitive property in an application change, these changes must be reported to the context mediator. The context mediator updates consumer-lists of affected nodes in the context assertion lattice and using this context assertion lattice generates a new list of Total-Suppliers-lists and Partial-Suppliers-lists for all new nodes in the characterizing environment of the non-primitive property in CDR of the application. The application should not request for data related to the non-primitive property until the context mediator returns a new set of Total-Suppliers-lists and Partial-Suppliers-lists to the set of new nodes entered into the application CDR.

Changes in Data Source's Data Semantics: Whenever the semantics of a non primitive property at a data source change, such changes will be reported to the context mediator. The context mediator updates source-lists of nodes in the context assertion lattice corresponding to the old and new contexts of the non-primitive property. Concurrently, the context mediator identifies the set of affected applications using consumer-lists attached to nodes in the context assertion lattice, evaluates new Total-Suppliers-lists and Partial-Suppliers-lists, and updates the CDR of each affected application. If a data-source changes the semantics of a non-primitive property, then it should not allow any application to access this property until the context mediator resolves all semantic mismatches by updating the CDRs of all affected applications.

In the above discussion, a mechanism for semantic interoperability between an application and a data-source with respect to a single non-primitive property is discussed. In fact, an application query may consist of one or more non-primitive properties and may have some primitive properties. In such a situation, identification of data-sources which can supply a semantically meaningful solution to an application query in a cost-effective manner is a challenging task. In the following section, we proposed an algorithm to identify a set of data-sources which can supply semantically meaningful data to an application query.

6 Query Processing

An application generates a query based on its data requirements. Each query consists of two parts: the target part and the qualification part. An application query can be interpreted as a request for the instances of the target properties within the context given by the qualification of the query. The semantics of the properties referred to in the query can be determined by the CDR of the qualification of the query. Therefore, one needs to identify semantic assertions which are true in the context given by the qualification part of the query and the CDR of the application.

The context of a non-primitive property in the query can be defined only if the query is well-formed at the application. We define a query to be well-formed with respect to a non-primitive property at an application, if the semantics of the non-primitive property in the query is derivable from the qualification of the query and the CDR of the application. In other words, the qualification of the query must be complete enough to determine the semantic assertions of a non primitive property referred to in the query. A well-formed query with respect to a non-primitive property at a data source can be defined in a similar manner. Once semantic assertions of properties referred to in the query are determined, then the query processor identifies a set of sites which can supply semantically meaningful solution to the query. In the following subsection, we provide an algorithm to identify a set of such sites.

6.1 Site-selection Algorithm

The process of site-copy selection is a mechanism for selecting a set of data sources for processing a given query if more than one candidate set are available. The site-copy selection algorithm in MERMAID [Te87] concentrated on optimization of query execution cost and did not consider semantic heterogeneity aspects during the selection of processing sites, which are instead resolved by the integrated schema in MERMAID architecture. The MERMAID algorithm selects the set consisting of the minimum number of sites to process the query out of all possible sets of candidate sites. The rationale behind the minimum number of sites is that, in general, the data communication requirements are likely to be minimum if the number sites involved in the query processing are minimum, which in turn reduces the overall query processing cost. Since there is no integrated schema in our context interchange architecture, the Site-copy selection algorithm must deal with semantic heterogeneity issues. The algorithm below consideres such issues during the selection of processing sites.

The following algorithm selects a set of candidate sites for a given query which can supply meaningful solution to the query. The Total-Suppliers-List and Partial-Suppliers-List used in the following algorithm are defined in the previous section.

```
**Candidate-sites**(query, CDR(application))
    {
      Let P be the set of primitive properties and X be the set of non-primitive
      properties referred to in the query;
    /* see Example-1 and Example-2 below */
      For each $p_i \in P$, let $SP_i$ be the set of data-sources where $p_i$ is available;
      Represent $SP_i$ in disjunctive normal form.
      For each $x_j \in X$, whose context is given by $n_{x_j}$
/* see Example-1 below */
{ if total-suppliers list of $n_{x_j}$ is not empty
    { let $SX_j$ denote the Total-Suppliers-List of $n_{x_j}$;
      Represent $SX_j$ in disjunctive normal form }
/* see Example-2 below */
else if Partial-Suppliers-List of $n_{x_j}$ is not empty
     { let $SX_j$ denote the collection of Partial-Support-List of $n_{x_j}$;
       Represent each $s_{jl} \in SX_j$ in disjunctive normal form
        and assign their conjunction to $SX_j$ }
}
 Set Candidate-Sites to $(\vee_{p_i \in P} SP_i) \wedge (\vee_{p_j \in X} SX_j)$;
 Translate the expression Candidate-Sites into disjunctive normal form;
 select the conjective term from Candidate-Sites which has minimum number of sites
 and if more than one such term exists then select any one of them
 and return the selected term.
}
```

Examples:

The following examples illustrate the selection of candidate sites to process a given query by taking semantic heterogeneity issues into consideration.

Example-1: Assume that the following query is received from the application ap1.

Select Trade-price

Where (Exchange=Nyse and Instrument-type=Equity).

The query is well-formed at the application ap1.

P = { Exchange, Instrument-type }

X = { Trade-price }

$SP_{Exchange} = db1 \vee db2 \vee db3$

$SP_{Instrument-Type} = db1 \vee db2 \vee db3$

$SX_{Trade-price} = db1$

Candidate sources for processing the complete subquery is given by

$(db1 \vee db2 \vee db3) \wedge (db1 \vee db2 \vee db3) \wedge (db1)$

The above expression is translated into disjunctive normal form as follows:

$(db1) \vee (db1 \wedge db2) \vee (db1 \wedge db3) \vee (db1 \wedge db2 \wedge db3)$

Since the first disjunction has minimum number of elements, Data-source db1 will be selected to process the query.

Example-2: Assume application ap3 uses the query:

Select Trade-price

Where Exchange=Nyse

The query is well-formed at the application ap3.

P= { Exchange }

X = { Trade-price }

$SP_{Exchange} = db1 \vee db2 \vee db3$

Since, Trade-price in the application ap3 has no complete solution at any single data-source. It has only partial solution at db1 and at db3.

$SX_{Trade-price} = (db1) \wedge (db3)$

Candidate sites to process the query is given by

$(db1 \vee db2 \vee db3) \wedge ((db1) \wedge (db3))$

The above expression is translated into disjunctive normal form as follows:

$(db1 \wedge db3) \vee (db2 \wedge db1 \wedge db3)$

Data-sources, db1 and db3 can be selected to process the query.

7 Conclusion

In this paper, we provided a lattice-based framework for the description of context knowledge for data-sources and applications. Since hierarchical nature is implicit among disparate contexts, the lattice model is ideally suited for the required representation. Our approach requires that a context data repository(CDR) of an application and another of a data-source be generated from the given set of context-lattices. In other words, all CDRs must share the same set of context-lattices. Context comparison and context evolution tasks can be handled efficiently using the lattice model. The lattice technique for representing context knowledge is more appropriate since it involves only set operations for their context comparison, unlike as rule based representation which uses inference mechanism to perform context comparison.

The context-assertion lattice presents a set of legal contexts and associated assertions which can be used as a reference set for a Database Administrator or an application developer to frame his or her CDRs. If the application developer has the option to choose between semantic assertions then he or she can use the existing context-assertion lattice to identify what kind of assertion would receive greater database support. Similarly a database designer can use the context-assertion lattice to fix semantic assertions in his or her data so that they can be utilized meaningfully by many applications.

Our model is powerful enough to accommodate semantic conversions during context interchange through the notion of node equivalence. If two nodes in a context assertion lattice are equivalent then all applications associated with one of the nodes can get semantically meaningful data from the data sources associated with the other node and vice versa.

We also derived conditions under which a source can supply meaningful data to an application. Under these conditions, the context-mediator maintains the consistency of the knowledge present in the application as well as the data-source CDRs. The Query-processor, using the knowledge present in the CDR of the application, can identify a set of data-sources which can supply meaningful solution to the application query. As such, the roles of context-mediator and the query-processor are separated. A context driven data-source selection algorithm for a given application query was also described.

Since the storage requirement for each context-assertion lattice can be large, the lattice must be pruned to its minimum size before it is physically stored. All nodes in a context-assertion lattice, which are not present in any CDR can be pruned from the lattice. Since the lattice is a directed acyclic graph, well defined storage representation schemes and algorithms to include a given node into the lattice and to search for a required node in the lattice are available.

Acknowledgments: The authors thank anonymous reviewers for their detailed suggestions.

M. P. Reddy recently joined Kenan Systems as a database architect. Earlier, he worked as Research Associate at the Sloan School of Management, Massachusetts Institute of Technology from 1991 November through 1994 January. He received his doctorate in computer science and masters in physics from the University of Hyderabad, Hyderabad, India. Prior to joining MIT, he worked as an Assistant Professor at the University of Hyderabad. At MIT, Dr. Reddy was active in several research projects at the Composite Information Systems Laboratory. He has published several articles in the areas of heterogeneous databases and data quality management. His current research interests include integration of heterogeneous databases, context interchange among heterogeneous information systems, knowledge discovery in databases, data quality management, and multidimensional databases.

A. Gupta is Co-Director of MIT's Productivity from Information Technology (PROFIT) Initiative and the first and only Senior Research Scientist at the Sloan School of Management, Massachusetts Institute of Technology. He received his Ph. D. in computer science from the Indian Institute of Technology, New Delhi, a masters degree in management from MIT and a Bachelors degree in electrical engineering from the Indian Institute of Technology, Kanpur.

Dr. Gupta serves on the Administrative Committee of the IEEE Industrial Electronics Society and has been assistant chairman for several IECON conferences.

Since joining MIT in 1979, he has been active in the areas of multiprocessor architectures, distributed and heterogeneous database systems, and automated reading of handwritten information. He has published more than 100 technical articles and papers, and produced seven books in these areas. He serves as an advisor to a number of leading international organizations.

References

[BT84] Y. J. Breitbart and L. R. Tieman. ADDS-heterogeneous distributed database system. In *Proceedings of Third International Seminar on Distributed Database Systems*, March 1984.

[Bur52] E. Burnswick, editor. *The Conceptual Framework of Psychology.* University of Chicago Press, Chicago, 1952.

[Coe77] T. R. Coe, W. C. Sarbin. Hypnosis from the standpoint of a contextualist. *Annals of the New York Academy of Science*, 296, 1977.

[DAT87] S. M. Deen, R. R. Amin, and M. C. Taylor. Implementation of a prototype for PRECI*. *Computer Journal*, 30(2), 1987.

[dK86a] J. de Kleer. An assumption based TMS. *Artificial Intelligence*, 28, 1986.

[dK86b] J. de Kleer. Extending the ATMS. *Artificial Intelligence*, 28, 1986.

[dK86c] J. de Kleer. Problem Solving with ATMS. *Artificial Intelligence*, 28, 1986.

[Doy89] R. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12, 1989.

[Ke88] V. Krishnamurthy and et al. IMDAS- An integrated manufacturing data administration system. *Data and Knowledge Engineering*, 3, 1988.

[LR82] T. Landers and R. Rosenberg. An overview of MULTIBASE. In *Distributed Data Bases*, pages 153–183. North Holland, 1982.

[Lyo81] J Lyons, editor. *Language, Meaning and Context.* Fontana Paperbacks, Great Britain, 1981.

[McD83] D. V. McDermott. Contexts and Data Dependencies: A Synthesis. *IEEE Transactions on Patteren Analysis and Machine Intelligence*, 5, 1983.

[RJPS89] M. Rajinikanth, G. Jakobson, and G. Piatetsky-Shapiro. On heterogeneous database integration: One year experience in evaluating CALIDA. In *Proceedings of Workshop on Heterogeneous Databases*, December 1989.

[Sho91] Y. Shoham. Varieties of Context. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: papers in honor of John McCarthy.* Academic Press, 1991.

[SM91] M. Siegel and S. Madnick. A Metadata Approach to Resolving Semantic Conflicts. In *Proceeding of the 17th International Conference on Very Large Data Bases*, September 1991.

[SSR92] E. Sciore, M. Siegel, and A. Rosenthal. Context interchange using meta-attributes. In *Submission to the 18th International Conference on Very Large Data Bases*, August 1992.

[Te87] M. Templeton and et al. Mermaid - a front-end to distributed heterogeneous databases. In *Proceedings of the IEEE*, volume 57, pages 695–708, 1987.

[Tem89] M. Templeton. Schema integration in mermaid. In *Position Papers: NSF Workshop on Heterogeneous Databases*, December 11-13, 1989.

[Tho88] G. M. Thomson, D. M. Davies. Introduction: Memory in Context: Context in Memory. In Davies G. M. and Thomson D. M., editors, *Memory in Context: Context in Memory.* John Wiley & Sons Ltd., 1988.